Adversarial Nanomunchers

Dennis Shasha

Omniheurist Course

Computer Science

Description

First we will discover basic nanomunchers. However, we will be playing adversarial nanomunchers which will be much more fun.

We had heard of Dr. Robert Hatchett. A world authority on bioterrorism and toxic waste disposal, he exuded a sense of ironic good humor at the over-hyped technology he heard about every day.

``Nanomachines in science fiction play the role of magic,'' he said with a smile. ``If one needs a super-flexible robot, a swarm of nanomachines fits the bill. If one needs extra-sensory perception, nanomachines can do that too. The forseeable reality however suggests that nanomachines have pico-intelligence.

``In fact the *nanomunching* machines we are planning to deploy for hazardous waste disposal will follow a trivial program of the form:

eat at start position
loop
if there is something to eat to the left then go there and eat it
if there is something to eat above then go there and eat it
if there is something to eat to the right then go there and eat it
if there is something to eat below then go there and eat it
end loop

``We abbreviate such a loop as left, up, right, down. Because the nanomuncher gets its nutrients from what it eats, it won't visit a node that it has already eaten or that another nanomuncher has partly eaten. The nanomuncher will not even go through that node. All it will do is keep following the loop on the graph it is assigned to until it can't continue.

``Note that the loop keeps its `program counter' position after each move. So, if you start at A and go left to B, you will first try to go up from B. If not possible, then right, if not possible, then down if not possible then left. If none are possible, then B is called a black hole and the nanomuncher will

disintegrate. The goal is to eat at every node in a given graph before entering a black hole. If you succeed, then you have `munched' the graph.

``You are allowed to choose the first node and to choose the order in the loop. So a loop might be right, up, left, down, for example. As long as the loop has only four distinct commands, you can chose any order e.g. left, up, down, right.

``You will be presented with a graph to be munched and your job is to determine how many synchronous nanomunchers you need and where and when they should start in order to munch the whole graph. The goal is to munch the whole graph using the minimum number of nanomunchers and, within that minimum, to use as little time as possible. Thus, a solution A is better than B if it requires fewer nanonmunchers. If they use the same number of nanomunchers, then A is better if A does this faster."

node(nodeid)
edge(nodeid, nodeid)

If several nanomunchers all land on the same node, then only one lives. The one that that moves up has highest living priority, then left, then down, then right. If a new nanomuncher lands on the same node as an existing one, then it dies on the spot (rookies always lose). If two or more new nanomunchers land on the same spot, then the architect's program decides which should live by uniform and random choice.

You may start some of your nanomunchers later than at the beginning, but then you are charged for the number of starters.

The Architect's Task

The architect gets a description of a grid which locations have eatable nodes and where there are edges. Each person will give you a list of nanomuncher descriptions consisting of when it starts where it starts its loop program.

For basic nanomunchers (which you might play for practice but which will not be part of the competition), the simulation proceeds as follows: At time zero, each nanomuncher that begins at time 0 eats its start position. Synchronously, each nanomuncher tries to move and the first place it can move to (has an edge to it and has not been eaten), it moves to. Death may happen. Nanomunchers may enter later. They act similarly.

For both basic and aversarial, remember to keep the program counter, so if the last move was the third in the loop, then the first attempt from the next position will be the fourth in the loop. For example, if the program is left, up, right, down and you succeeded in moving using right, then you will next try down then left then up....

The simulation for adversarial nanomunchers is the same except there are two sides. A red side and a blue side. Each side gets to use k (> 1) nanomunchers in total (determined on the night of the competition), any subset of which may enter at any moment. The simulation must keep track of which side munches the most nodes. All moves are simultaneous as above and with the collision rules from above. Here is a movie of what a competition looks like (from 2013).

We'll have a tournament. There will be a bit more than 100 nodes in a 20 (x ranges from 0 to 19) by 10 (y ranges from 0 to 9) grid and some number of edges. The nodes will be in the format: nodeid,xloc,yloc and the edges: nodeid1,nodeid2. Edges will always be listed from smaller nodeid to larger, though all edges

are two-way. Here is some typical data . Here is some more data. .

You might also find the following reference implementation description helpful. Here is the 2008 code that the above was built from with some documentation.